Cool new stuff with Scikit-Learn 0.19 – 0.20-dev

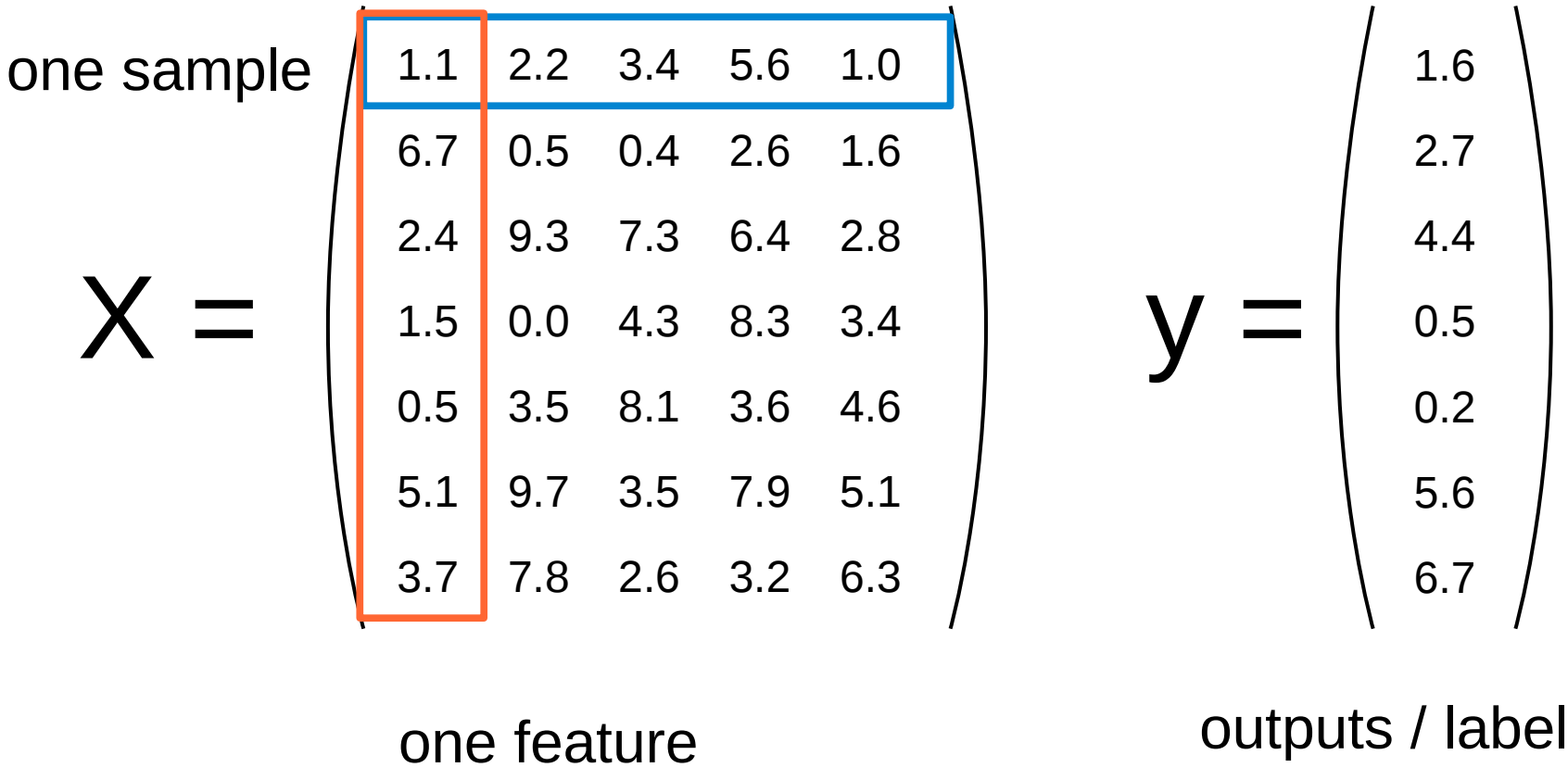
Andreas Müller
Columbia University, scikit-learn

Alfred P. Sloan
FOUNDATION

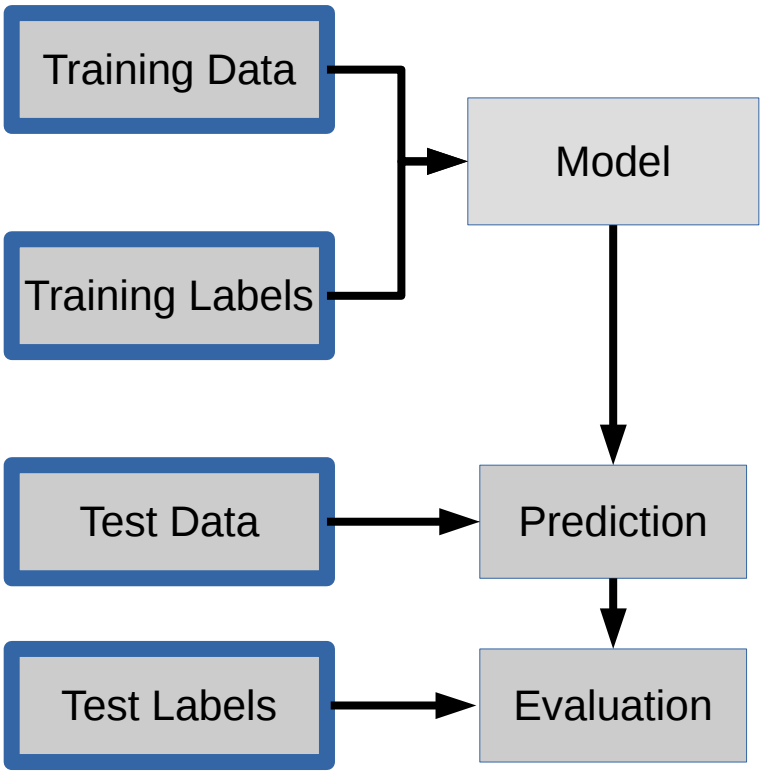
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Basic API review

Representing Data



```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



Basic API

```
estimator.fit(X, [y])
```

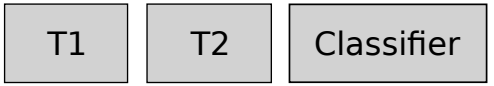
`estimator.predict`	`estimator.transform`
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Cross -Validated Grid Search

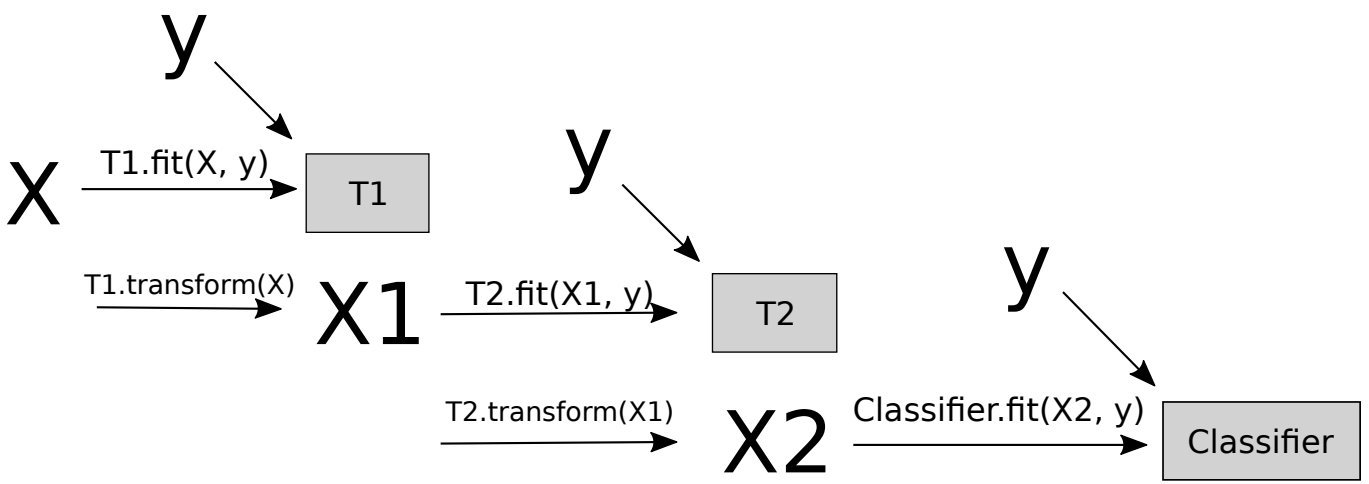
```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```

Pipelines

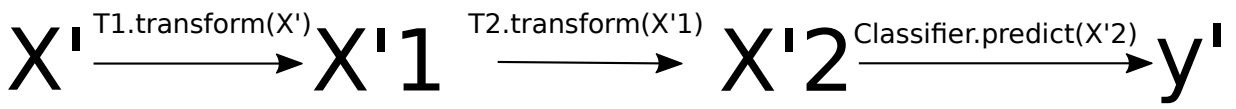
pipe = make_pipeline(T1(), T2(), Classifier())



pipe.fit(X, y)



pipe.predict(X')



Pipelines

```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), SVC())
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(StandardScaler(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

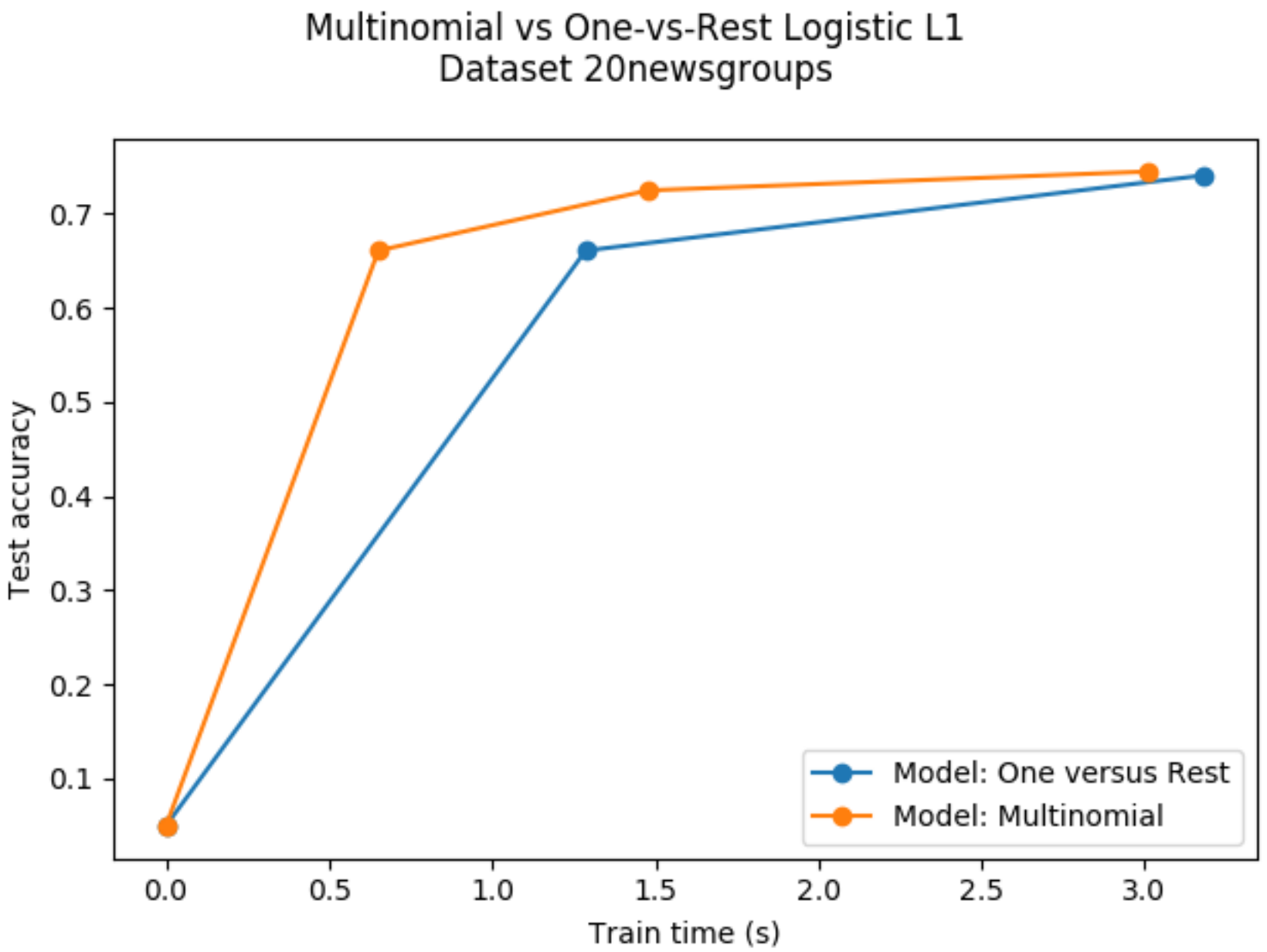
0.19

(August 2017)

Caching Pipeline

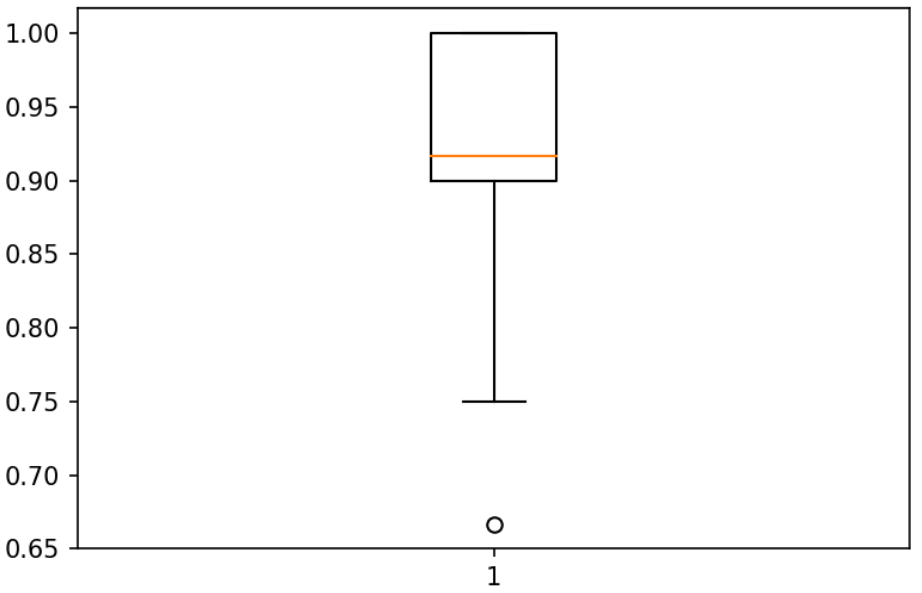
```
from tempfile import mkdtemp
from shutil import rmtree
from sklearn.externals.joblib import Memory

# Create a temporary folder to store the transformers of the pipeline
cachedir = mkdtemp()
memory = Memory(cachedir=cachedir, verbose=10)
cached_pipe = Pipeline([('reduce_dim', PCA()),
                        ('classify', LinearSVC())],
                       memory=memory)

# This time, a cached pipeline will be used within the grid search
grid = GridSearchCV(cached_pipe, cv=3, n_jobs=1, param_grid=param_grid)
digits = load_digits()
grid.fit(digits.data, digits.target)

# Delete the temporary cache before exiting
rmtree(cachedir)
```

SAGA

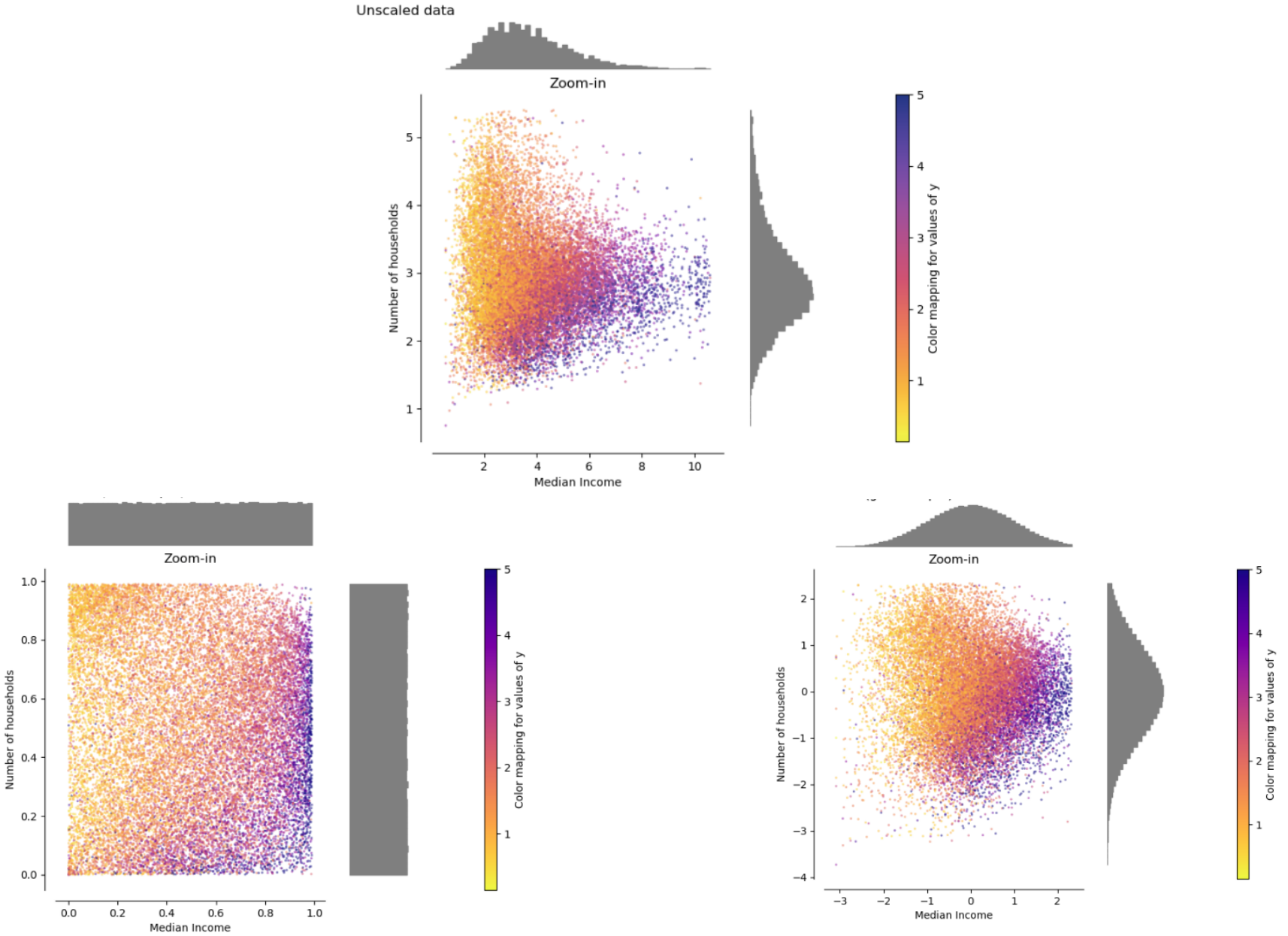

Multinomial vs One-vs-Rest Logistic L1
Dataset 20newsgroups
Test accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Train time (s)
Model: One versus Rest
Model: Multinomial

Repeated KFold

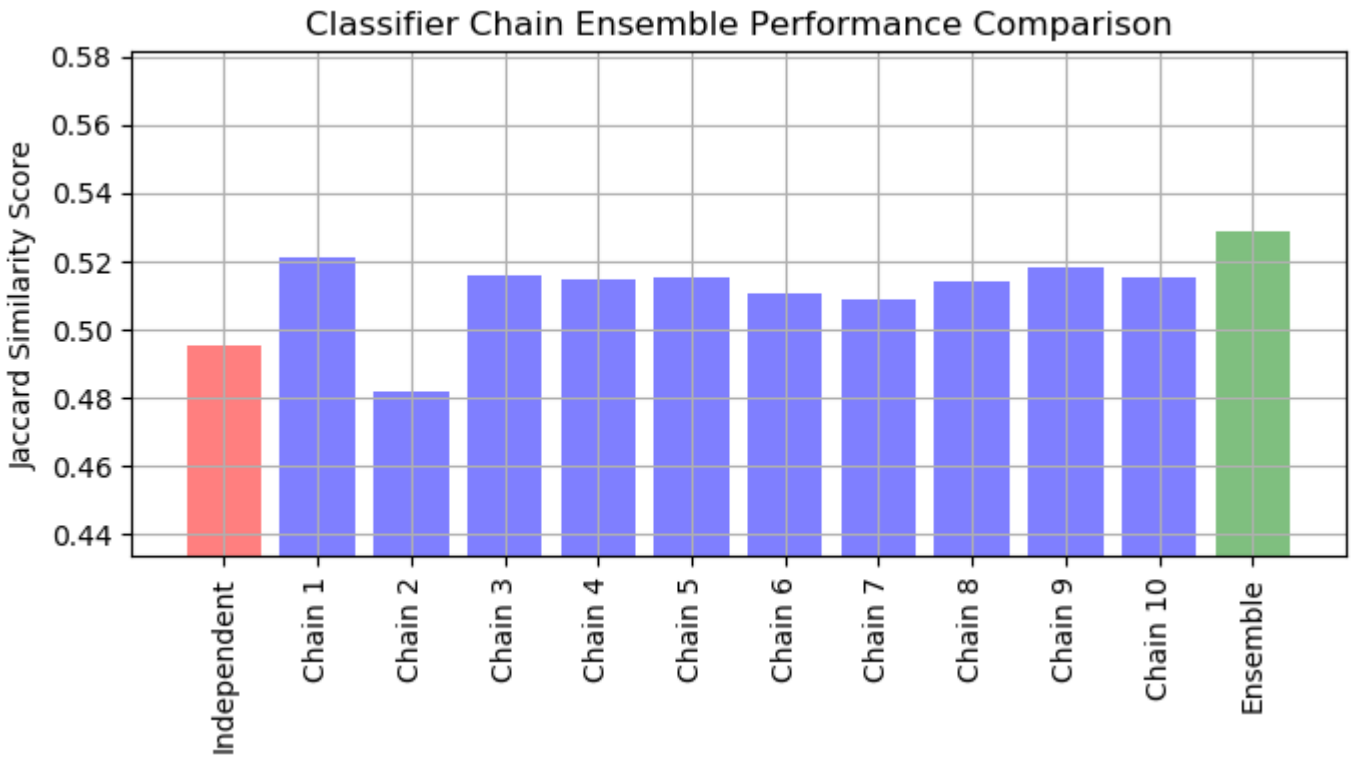
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold, RepeatedStratifiedKFold

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=42)
scores = cross_val_score(LogisticRegression(), X_train, y_train, cv=cv)
plt.boxplot(scores);
```



QuantileTransformer

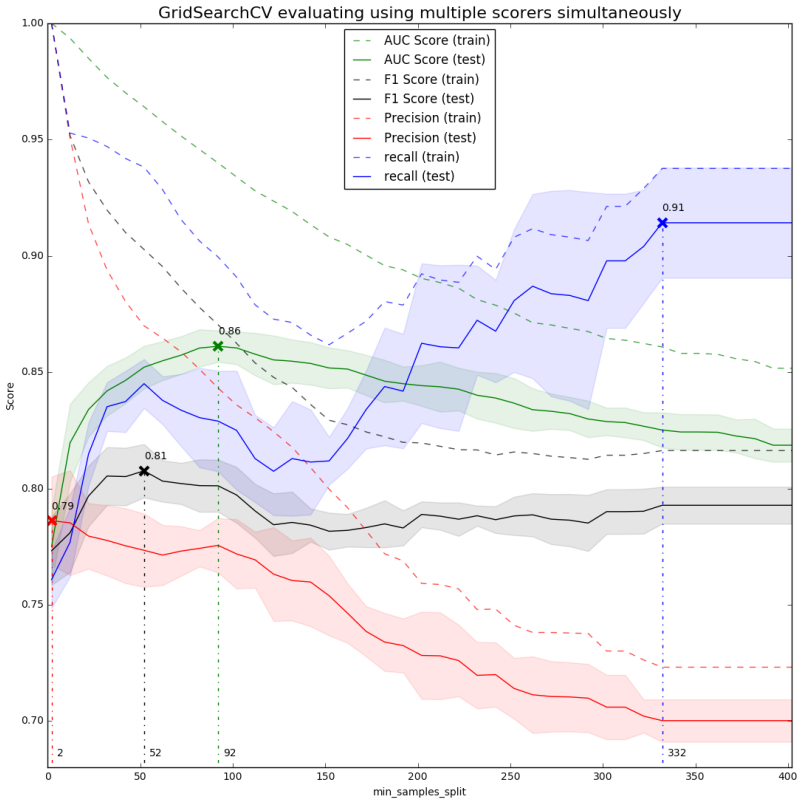


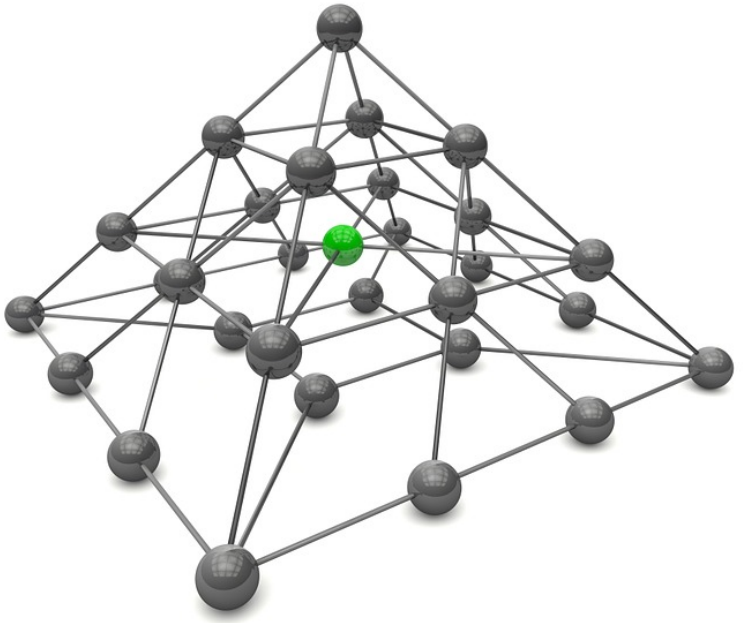
ClassifierChain



Multiple Metric Search

```
scoring = {'AUC Score': 'roc_auc', 'Precision': make_scorer(precision_score),
           'Recall': 'recall', 'F1 Score': 'f1'}

# Multiple metric GridSearchCV, best_* attributes are exposed for the scorer
# with key 'AUC Score' ('roc_auc')
gs = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid={'min_samples_split': range(2, 403, 10)},
                  scoring=scoring, cv=5, refit='AUC Score')
gs.fit(X, y)
```





Around Scikit-learn

Joblib Distributed Backend

```
import distributed.joblib
# Scikit-learn bundles joblib, so you need to import from
# `sklearn.externals.joblib` instead of `joblib` directly
from sklearn.externals.joblib import parallel_backend
from sklearn.datasets import load_digits
from sklearn.grid_search import RandomizedSearchCV
from sklearn.svm import SVC
import numpy as np

digits = load_digits()

param_space = {
    'C': np.logspace(-6, 6, 13),
    'gamma': np.logspace(-8, 8, 17),
    'tol': np.logspace(-4, -1, 4),
    'class_weight': [None, 'balanced'],
}

model = SVC(kernel='rbf')
search = RandomizedSearchCV(model, param_space, cv=3, n_iter=50, verbose=10)

with parallel_backend('dask.distributed', scheduler_host='localhost:8786'):
    search.fit(digits.data, digits.target)
```

Distributed Random Forests

```
Dataset statistics:
===================
number of features:       54
number of classes:        2
data type:                float32
number of train samples:  522911 (pos=332178, neg=190733, size=112MB)
number of test samples:   58101 (pos=36994, neg=21107, size=12MB)
```

```
Classification performance:
===========================
Classifier   train-time test-time error-rate
--------------------------------------------
RandomForest, threading 35.2176s 0.5228s 0.0296
RandomForest, dask.distributed 11.4347s 3.0020s 0.0296
ExtraTreesClassifier, threading 33.4432s 0.7229s 0.0325
ExtraTreesClassifier, dask.distributed 16.7369s 6.5222s 0.0325
```

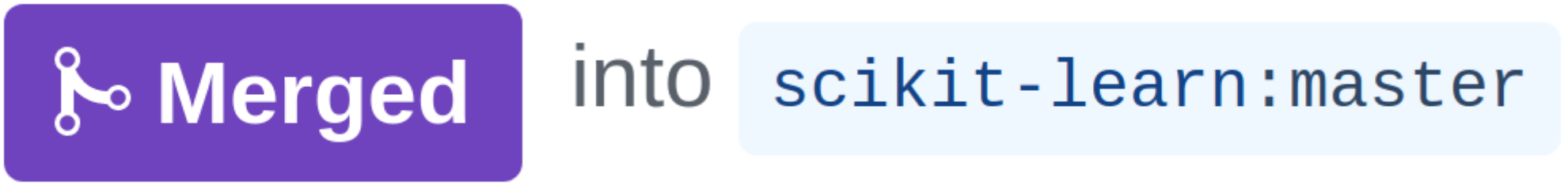
Uses scatter!
https://github.com/dask/distributed/pull/1022#issuecomment-297550998

dask_searchcv – lazy pipelines!

```
from sklearn.pipeline import Pipeline

logistic = linear_model.LogisticRegression()
pca = decomposition.PCA()
pipe = Pipeline(steps=[('pca', pca),
                       ('logistic', logistic)])


#Parameters of pipelines can be set using '__' separated parameter names:
grid = dict(pca__n_components=[50, 100, 150, 250],
            logistic__C=[1e-4, 1.0, 10, 1e4],
            logistic__penalty=['l1', 'l2'])

# from sklearn.grid_search import GridSearchCV
import dask_searchcv as dcv

estimator = dcv.GridSearchCV(pipe, grid)

estimator.fit(X, y)
```

0.20

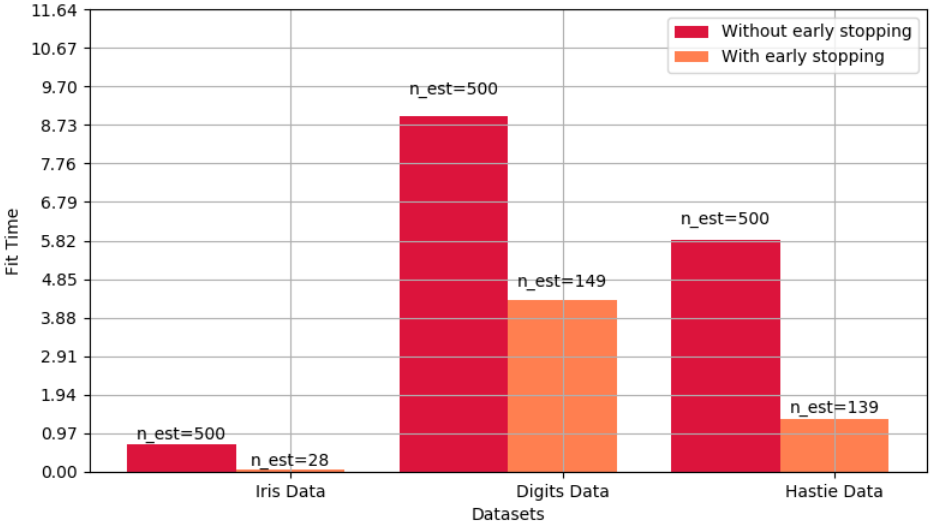
("early" 2018)


Merged
into
scikit-learn:master

Gradient Boosting Early Stopping

```
GradientBoostingClassifier(n_estimators=n_estimators,
                           validation_fraction=0.2,
                           n_iter_no_change=5, tol=0.01,
                           random_state=0)
```

Activated by “n_iter_no_change”

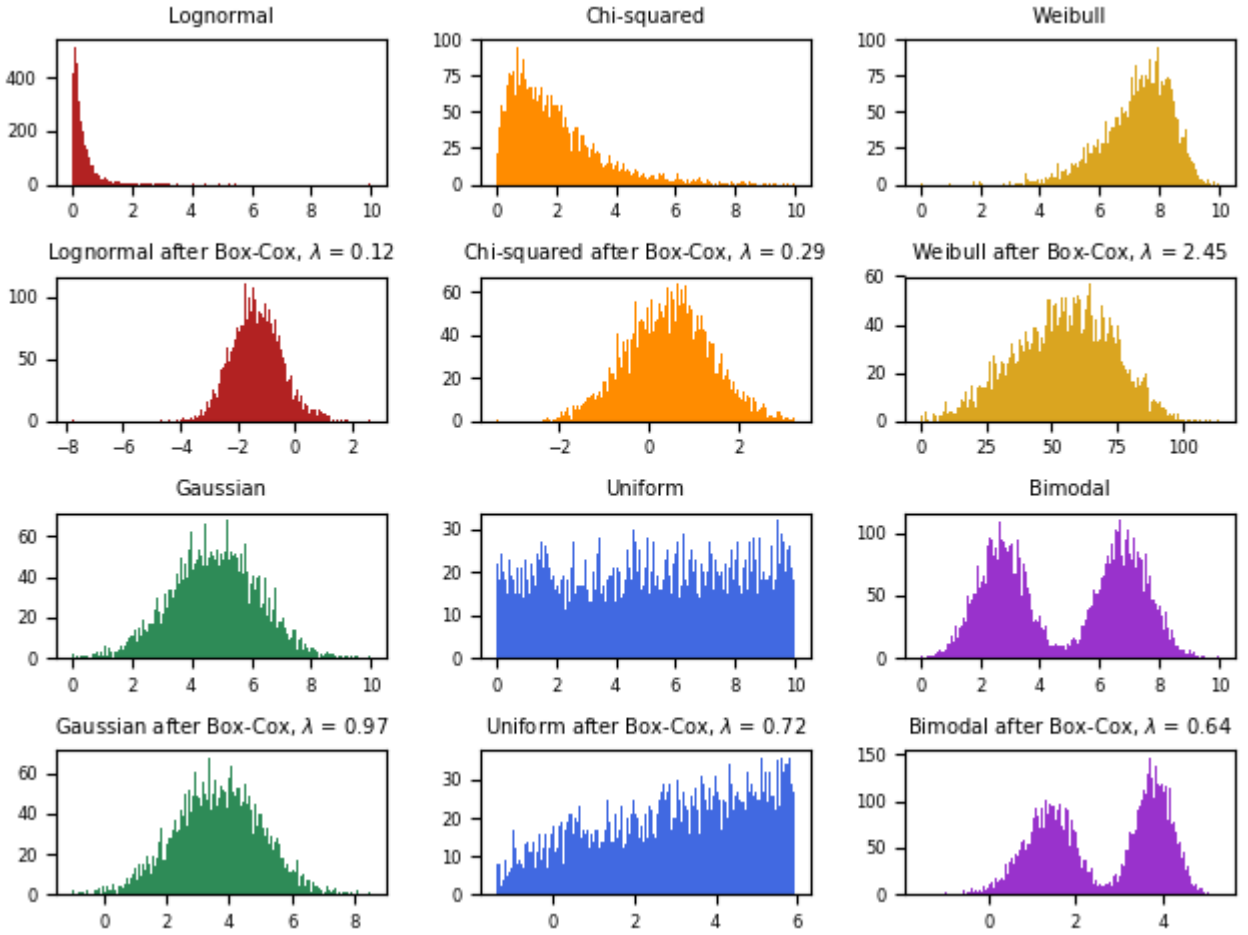


CategoricalEncoder

Extends / replaces OneHotEncoder
to work on text and integer data!

```
>>> enc = preprocessing.CategoricalEncoder()
>>> X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe', 'uses Firefox']]
>>> enc.fit(X)
CategoricalEncoder(categories='auto', dtype=<... 'numpy.float64'>,
          encoding='onehot', handle_unknown='error')
>>> enc.transform([['female', 'from US', 'uses Safari'],
...                ['male', 'from Europe', 'uses Safari']]).toarray()
array([[ 1.,  0.,  0.,  1.,  0.,  1.],
       [ 0.,  1.,  1.,  0.,  0.,  1.]])
```

PowerTransformer



Right now: Box-Cox transform
before release: also Yeo-Johnson

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

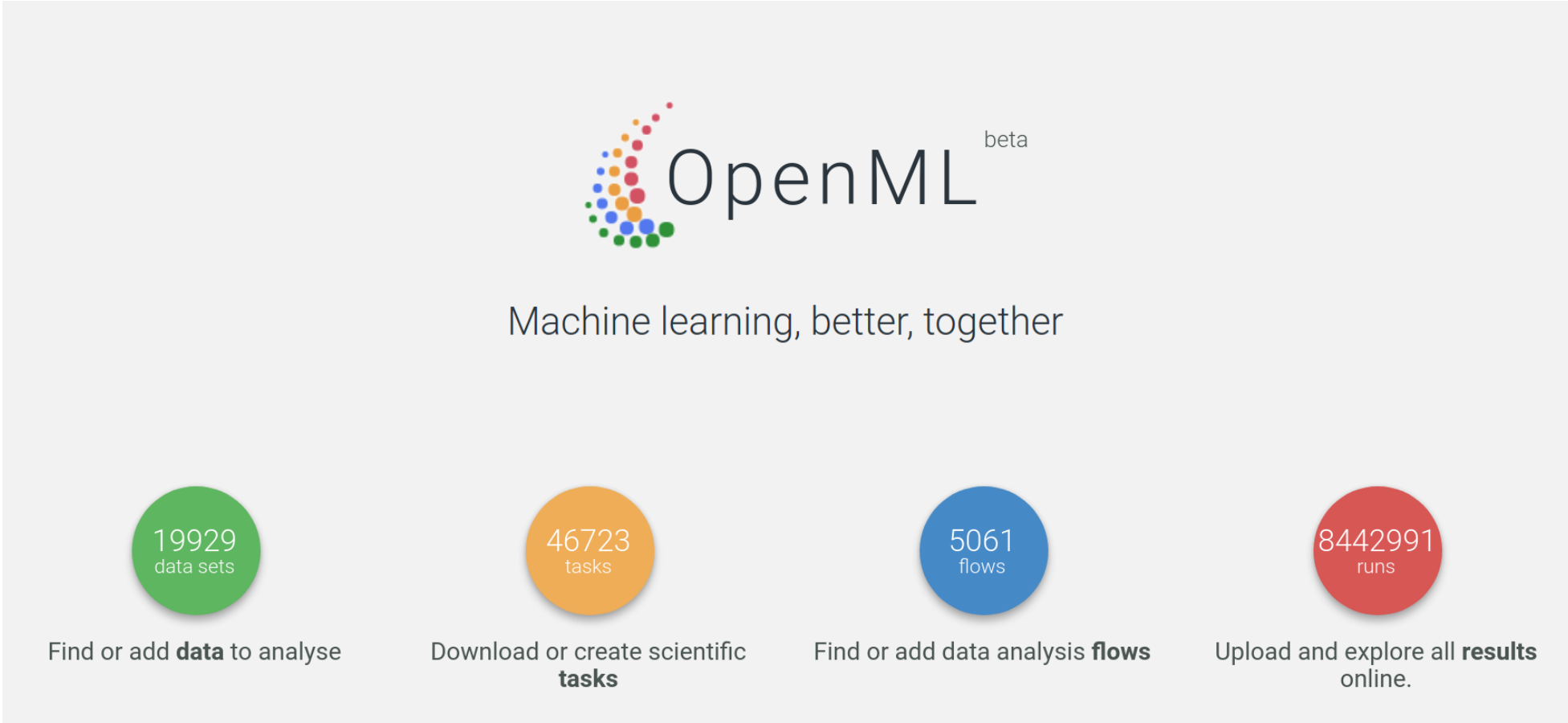
Work In Progress

ColumnTransformer

```
>>> from sklearn.experimental import ColumnTransformer
>>> from sklearn.preprocessing import Normalizer
>>> union = ColumnTransformer(
...     [("norm1", Normalizer(norm='l1'), [0, 1]),
...      ("norm2", Normalizer(norm='l1'), slice(2, 4))])
>>> X = np.array([[0., 1., 2., 2.],
...               [1., 1., 0., 1.]])
>>> # Normalizer scales each row of X to unit norm. Therefore, a separate
>>> # scaling is applied for the two first and two last elements of each
>>> # row independently.
>>> union.fit_transform(X)
array([[ 0. ,  1. ,  0.5,  0.5],
       [ 0.5,  0.5,  0. ,  1. ]])
```

Particularly userful to treat different types of data within the same dataset (continuous, categorical, text, ...)

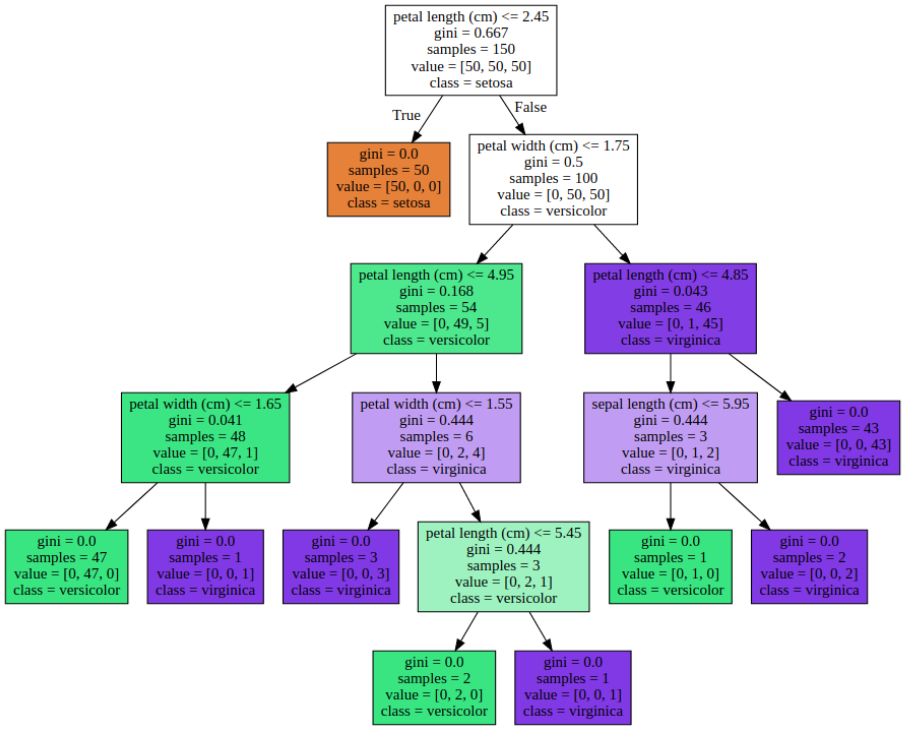
OpenML Dataset Loader



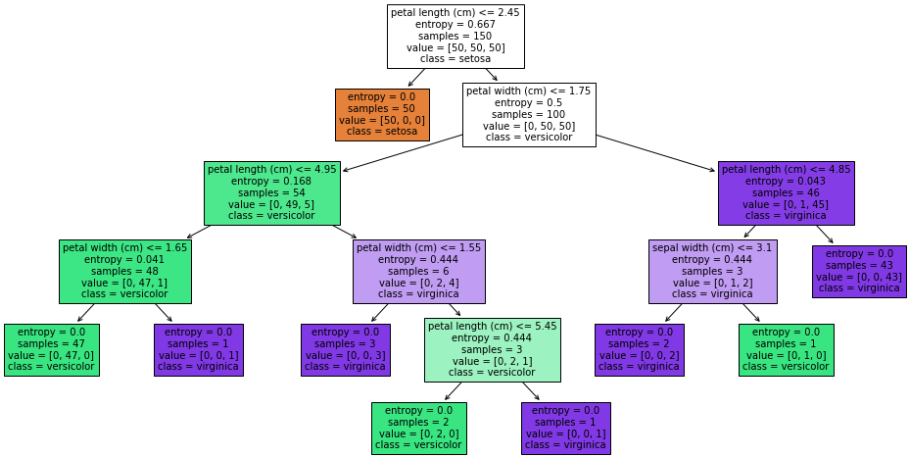

```
>>> from sklearn.datasets import fetch_openml
>>> mice = fetch_openml('miceprotein', version=4, data_home=custom_data_home)
```

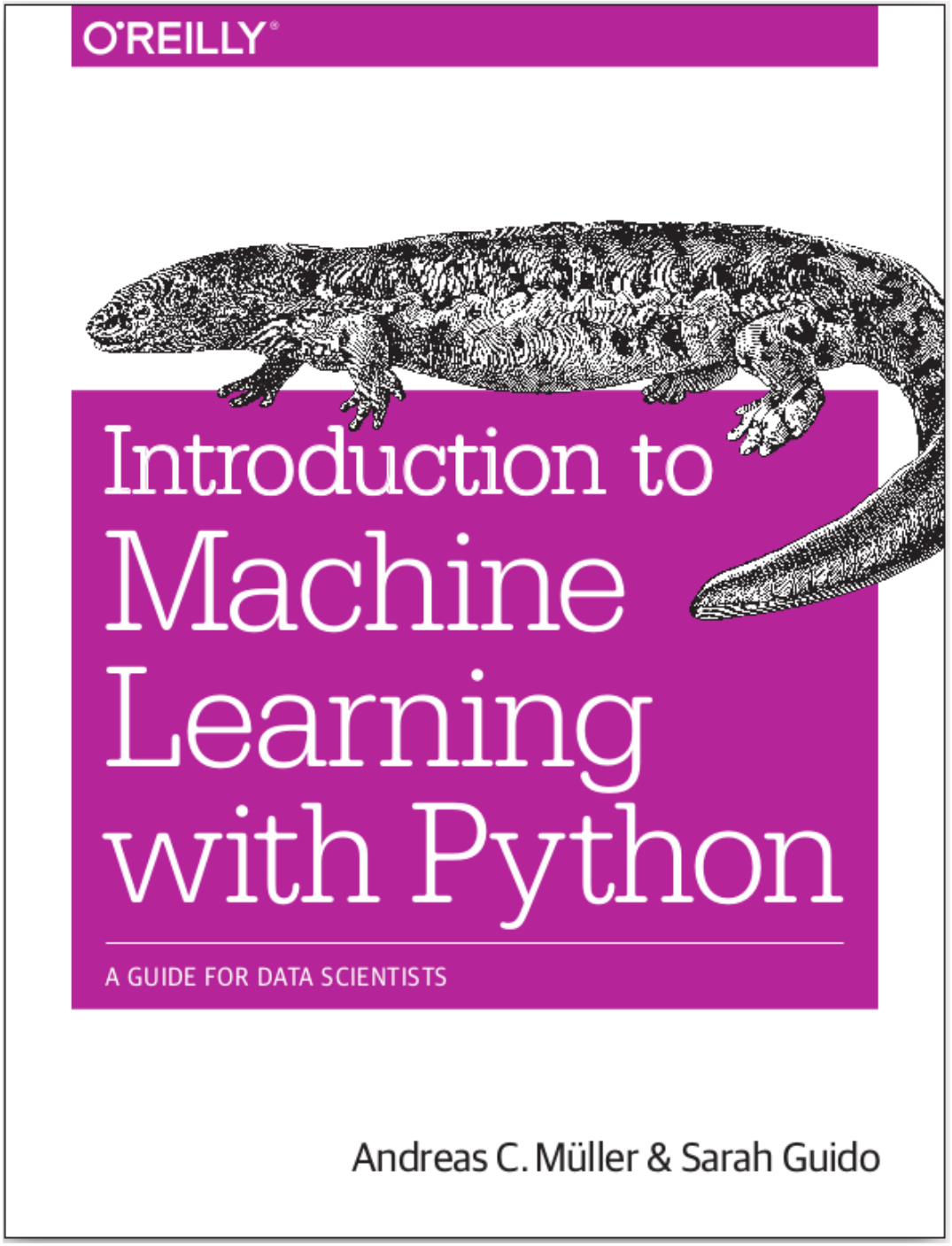
Matplotlib based tree plotting

Graphviz



Our own!





amueller.github.io

@amuellerml

@amueller

t3kcit@gmail.com